Check Game artificial intelligence based using reinforcement learning.pdf

by Arta Moro Sundjaja

Submission date: 25-Jan-2019 10:38AM (UTC+0700)
Submission ID: 619532227
File name: e_artificial_intelligence_based_using_reinforcement_learning.pdf (415.37K)
Word count: 4512
Character count: 22821

Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 50 (2012) 555 – 565

Procedia Engineering

www.elsevier.com/locate/procedia

International Conference on Advances Science and Contemporary Engineering 2012 (ICASCE 2012)

Game Artificial Intelligence Based Using Reinforcement Learning

Albertus Agung[a], Ford Lumban Gaol[b]

[a,b]Bina Nusantara University, Jl. Kebon Jeruk Raya 27, Kebon Jeruk Jakarta Barat 11530, Indonesia

Abstract

Research on artificial intelligence in the game creates an interest study in the world of artificial intelligence. There are two types of learning of reinforcement learning is used in the study of learning mechanisms. Through the analysis of simulation results a simple game application has been made to make a conclusion. Learning itself is provided with a variety of arenas as environment of agent who will be learning. From this research began by using two methods of learning Q-Learning and Sarsa. Comparison cleverness made before and after the agent was given intelligence drawn from these results that can create an inference that can be described in terms of type arena and in terms of learning speed.

Keyword: game, artificial intelligence, reinforcement learning, Q-Learning, Sarsa.

Keywords: Type your keywords here, separated by semicolons ;

1. Introduction

Artificial Intelligence gives a great taste as an important role in video games. It gives a chance for player not to play alone. Therefore the purpose to create the AI in computer games nowadays is to give a real experience for a player to play with human-like thinking abilities.

Game itself has a lot of variations that most of the experts prefer to focus on one specific research. Here are a lot of variations in video games. This table (TABLE 1[3]) shows us a short brief about variations in game.

1877-7058 © 2012 Elsevier B.V...Selection and peer-review under responsibility of Bin Nusantara University
doi:10.1016/j.proeng.2012.10.060

Table 1.1The categories of digital games

Genre	Examples
Classic board	Chess, Checkers
Adventure	Bonji's Adventures in Calabria
Team Sports	RoboCup Soccer
Real-time Individual	Bilestoad, Space Invaders
Real-time God	SimCity
Discrete Strategy	Freeciv
Real-time Strategy	Wargus

Although there are so many categories of games, the AI in all of the games is usually a critical point to discuss. Player often to meet some bugs happened in each of the game they play or player feel bored that the AI does only static movements. Therefore, it is of interest to find the best behavior and policy for AI in games.

Reinforcement Learning (RL) is one of the unsupervised machine learning methods in the area of artificial intelligence [2]. The concept of "trial-and-error" and the result of each "trial-and-error" action will be saved as a "delay reward". The ultimate goal of reinforcement learning is to give the machines human-like thinking and abilities.

Reinforcement Learning can be described as learning what to do, how to map situations to actions, so as to maximize a numerical reward signal [3]. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them.

The general aim of Machine Learning is to produce intelligent programs, it is called agents. Agent learns by interacting with its environment and observing the result of these interactions. This activity is related to what we, human, learn. Human have a direct sensor-motor connection to our environment, meaning human can perform actions and witness the results of these actions on the environment. In short word, it is commonly known as "cause and effect".

This research has a purpose to get particular data about the learningagent. Of course, we need to investigate learning method to do for the simulation result. Towards this goal, this paper narrows investigation to compare learning mechanism (i.e., Q-learning [Watkins and Dayan, 1992] and Sarsa [Rummery and Niranjan, 1994].

This paper is organized as follows. The next section starts by briefly explaining the simple application game, as an example for agent's environment. The third section discuss about the comparison of the result. The final section summarizes our conclusion.

2. Related Field Research

Yung-Ping Fang dan I-Hsien Ting wrote a research named "Applying Reinforcement Learning for the AI in a Tank-Battle Game". They divided the case into several points; Tank game: as first initialization to do experiments. They used game engine to create a Tank game. Reinforcement learning was applied as the algorithm for the Tank. Fuzzy logic was used as a concept to improve the performance of reinforcement learning. They do some experiments based on their research to find the best fuzzy functions.

AleksandarMicić, DavíðArnarssondanVignirJónsson wrote a research named "Developing Game AI For the Real-Time Strategy Game Starcraft". They implemented one of the unit in the game, called dragoon, and the other unit called zealot is controlled by AI. They have different characteristics, for example: Dragoon is a ranged unit and Zealot is a melee unit. The reinforcement learning which is implement in Dragoon unit explain us to do a great survivability. The survivability will be achieved if the Dragoon can both attack and move out from Zealot. This research calculates a difficult balancing between attack and move.

Keiki Takadama and Hironori Fujita in a research called "Q-Learning and Sarsa Agents in Bargaining Game" told us about the different of reinforcement learning between Q-learning and Sarsa. They found that the result between Q-learning and Sarsa are similar in several point of view.

3. Simple Application Game: Escape Labyrinth

This research is inspired from Tim Eden, Anthony Knittel, and Raphael van Uffelen application, about Cat and Mouse [4]. This research is also has the purpose to directly get the result and compare with any other related research. Just as written before about the common idea in reinforcement learning is known as cause and effect, and this is the key to building up knowledge of the environment later.

The cause and effect translated into the following steps for an RL agent:

1. The agent observes an input state
2. An action is determined by a decision making function (policy)
3. The action is performed
4. The agent receives a scalar reward or reinforcement from the environment
5. Information about the reward given for the

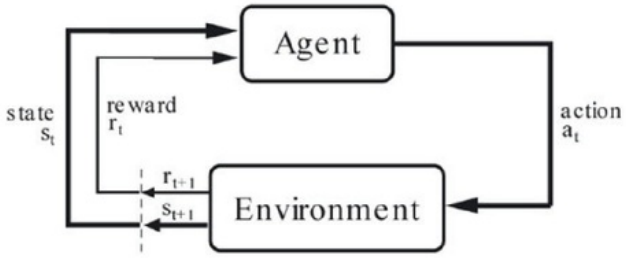

Fig. 2.1Agent-Environment interaction in reinforcement learning [Sutton and Barto, 1998]

The conceptual model of reinforcement learning fits very nicely to that of agents as used in 2apl (A Practical Agent Programming Language) [5]. In both cases there is a decision making agent who operates in an unknown environment at which it tries to accomplish a task. The perceive-think-act loop is present in both models. The only addition to the learning model is receiving a feedback from the environment on the effectiveness of the performed actions in relation to the task we are learning.

As term learning is related to reinforcement learning. We need to describe about supervised and unsupervised learning. Supervised learning involves learning with some supervision from an external source (i.e. a teacher) whereas unsupervised learning does not. This learning is similar to a student taking an exam, having it marked and then being shown which questions they answered incorrectly. Students are able to learn to answer the questions successfully after they learnt the correct answers. Unsupervised learning is far different from supervised learning where learning has been done without supervision. An example of the unsupervised learning is someone learning to juggle by themselves. The person will start by throwing the balls and attempting to catch them again. After a lot of failed handling, they will adjust their technique to make the balls still keep in the air.In short words, reinforcement learning is a form of unsupervised learning. Agent is asked to learn by receiving reward from its environment, without any form of supervision.

In order to do cause and effect in reinforcement learning, agent need to do exploration and exploitation. This is one of the interesting problems when using Reinforcement Learning. We need to set the tradeoff between exploration and exploitation. Agent need to try a certain action in the past and got a decent reward, and then repeating this action is going to reproduce the reward. The agent also needs to exploit what it knows to receive a reward while trying other possibilities may produce a better reward. That's why it needs a method to make it balance between exploration and exploitation so that the agent will learn successfully.

Reinforcement Learning offers to solve a lot of different problems. The main problem served in this research is talking about agent in simple games. The common knowledge about what reinforcement learning help in this part is defined the best move to make in a game. There are a lot of possible ways for agent to create its movement. Reinforcement Learninghelps to avoid large number of hard coded rules to specify the rules. Agent learns simply by playing on its environment.

In order to realize the step by step reinforcement learning agent, we need to determine about possible action.State-action pair functions that create estimation about the preferable action for agent, is called Value Functions. In this research, we used this following notation[3]:

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} \mid s_{t=s}\},$$

– the value of a state s under policy π. The expected return when starting in s and following πthereafter.

$$Q^{\pi}(s, a) = E_{\pi}\{R_t \mid s_t = s, a_t = a\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} \mid s_{t=s}, a_t = a\},$$

–the value of taking action a in a state s under a policy π. The expected return when starting from s taking the action a and thereafter following policy π.

After Value Functions are defined correctly, we need methods which can be used to estimate these value Functions. It is called Temporal Difference Learning that can help do the estimation. If the value functions were calculated without estimation, agent would need to wait until the final reward was received before any state-action pair values can be updated. When final reward was received, the path taken to reach the final state would need to be traced back and each value will be updated. This Temporal Difference Learning can be used by this following notation [3]:

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t - V(s_t)],$$

- $s_t$is the state visited at time t, $R_t$is the reward after time t and α is a constant parameter

Estimate of the final reward is calculated at each state and the state-action value updated for every step of the way with Temporal Difference methods:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)],$$

- $r_{t+1}$is the observed reward at time $t+1$.

Because the value is updated partly using an existing estimate and not a final reward, this temporal difference method is called a "bootstrapping".

There are two kinds of policy learning that we are going to describe in this research. The first one is On-Policy Temporal Difference method learning. It learns the value of the policy that is used to make decisions.The value functions are updated using results from executing actions determined by some policy. These policies are usually "soft" and non-deterministic. Soft means ensures there is always an element of exploration to the policy. The policy is not so strict that it always chooses the action that gives the most reward. There are three common polices are used: ε-soft, ε-greedy and softmax. Off-Policy methods can learn different policies for behavior and estimation. Similar with the one explained before, the behavior policy is usually "soft" so there is no sufficient exploration going on. This policy algorithm can update the estimated value functions using hypothetical actions, those which have not actually been tried. This is in contract to on-policy methods which update value functions based strictly on experience. What this means is off-policy algorithms can separate exploration from control, and on-policy algorithms cannot. In other words, an agent trained using an off-policy method may end up learning tactics that it did not necessarily exhibit during the learning phase.As mentioned before, the aim of policies is to balance the trade-off between exploitation and exploration, by not always exploiting what has been learnt so far.

ε-greedy–this policy allows the agent to do action with the highest estimated reward. This method ensures that if enough trials are done, each action will be tried an infinite number of times, thus ensuring optimal actions are discovered.

ε-soft–very similar to ε-greedy. The best action is selected with probability 1 –εand the rest of time a random action is chosen uniformly.

Softmax–one drawback of ε-greedy and ε-soft is that they select random actions uniformly. The worst possible action is just as likely to be selected as the second best. Softmax remedies this by assigning a rank or weight to each of the actions, according to their action-value estimate. A random action is selected with regards to the weight associated with each action, meaning the worst actions

are unlikely to be chosen. This is a good approach to take where the worst actions are very unfavourable.There is still no certainty that which of these policies produce the best results overall.

We try to compare two learning methods: Q-Learning and Sarsa in a simple application game. Q-Learning is an Off-Policy algorithm for Temporal Difference learning. It can be proven that given sufficient training under any ε-soft policy, the algorithm converges with probability 1 to a close approximation of the action-value function for an arbitrary target policy. Q-learning learns the optimal policy even when actions are selected according to a more exploratory or even random policy. Here is the procedural form of the algorithm:

Initialize Q(s, a) arbitrarily
Repeat (for each episode) :
    Initialize s
    Repeat (for each step of episode):
        Choose a from s using policy derived from Q
            (e.g., ε-greedy)
        Take action a, observe r, s'
        $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max\alpha, Q(s', a') - Q(s, a)]$
        $s \leftarrow s'$;
    until s is terminal

Parameters used in this process as follows:

α – learning rate, value between 0 and 1.With set to 0 means that Q-values will never be updated, then there is no any thing to be learned.To set it at such a high value of 0.9 means that learning can be accomplished quickly.

γ – discount factor, value between 0 and 1.This case illustrates that the future awards is worth less than the immediate rewards. The discount factor need to be set less than 0 for the algorithm to converge.

$\max_{\alpha}$- maximum reward that can be obtained at the following state. For example the reward for taking the optimal action thereafter. Approach procedure can be translated as follows:

1. Initialize the Q-values table, Q(s, a).
2. Observe the current state, s.
3. Choose an action, a, for that state based on one of the action selection policies.
4. Take the action, and observe the reward, r, as well as the new state, s'.
5. Update the Q-value for the state using the observed reward and the maximum reward possible for the next state. The updating is done according to the forumla and parameters described above.
6. Set the state to the new state, and repeat the process until a terminal state is reached.

Now let's take a look at Sarsa procedures. The Sarsa algorithm is an On-Policy algorithm for TD-Learning. The main difference between Sarsa and Q learning is the maximum reward for the next state is not required to be used to update the Q-values.In contrast, the new action, and as reward, selected using the same policies that determine the original action. Sarsa name actually comes from the action to perform the update by using a quintuple Q (s, a, r, s ', a'). Where: s, a are the original state and action, r is the reward observed in the following state and s', a' are the new state-action pair.form of the

sarsaalgorithm as follows :as you can see, there are two selection steps necessary action to obtain the next state-action pair along with the first. The parameters a and g have the same meaning as on the Q-Learning

Initialize Q(s, a) arbitrarily
Repeat (for each episode) :
Initialize s
Choose a from s using policy derived from Q
(e.g., ε-greedy)
Repeat (for each step of episode):
Take action a, observe r, s'
Choose a' from s using policy derived from Q
(e.g., ε-greedy)
Take action a, observe r, s'
$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
$s \leftarrow s'$; $a \leftarrow a'$;
until s is terminal

Applications used in this study are composed of individual components of the files related to one another. Making an application is inspired from the study of reinforcement learning by Tim Eden, Anthony Knittel, and Raphael van Uffelen. In which case they make a study to introduce reinforcement learning in game. The main structure in this study is divided into 2 parts:

1. Module reinforcement learning
   a. RLearner –reinforcement learning algorithm.
   b. RLPolicy. – Using Q-values table to decide the best action.
   c. RLWorld – Interface for RL world.
   d. World – World game implementation
2. Application Module Control
   a. Game – The game process
   b. RLController – Set the reinforcement learner, RLearner.
   c. sampleWorlds – Choice of arena in the game arena.
   d. boardObject – drawing object used in the application
   e. boardPanel – Organize drawing objects contained in the application panel.
   f. App - Class contains the main application that combines the other class.

4. Result

In the report results of this study, some of the arena (World) as a facility to support the learning outcomes has been conducted. Differences in place of the agent determine the outcome is different from the others. In the application there are two types of action the implementation of policies (Softmax and Greedy) and two types of learning method (SARSA and Q-Learning). Each of them will be paired one by one and carried out tests to find out the results after using the option. In this case it can be said that the site could affect the efficient agent in doing the learning. Here are a few reports of the arena that has been tested. In the training phase, the current study the same set of parameters to be used in the writing of this research. These data parameters are used to support learning in the later agent.

Death Penalty	: 100
Goal Reward	: 50
Alpha	: 1.0
Gamma	: 0.1
Epsilon	: 0.1
Action Selection Method	: Softmax/Greedy
Learning Method	: SARSA/Q-Learning
Epochs(Game)	: 50000

4.1. All World Results

A number of experiments carried out in the end gives the results of the agent behavior in the game are presented in several different arenas.

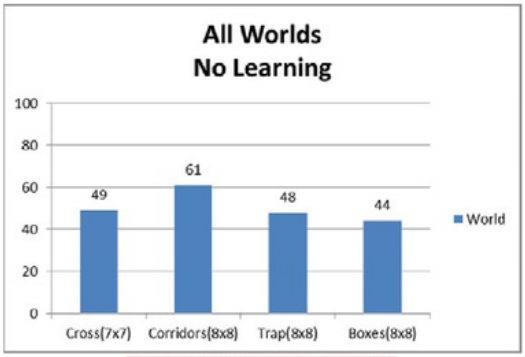

Fig. 2.15 All Worlds No Learning

The experimental results without using the whole arena of learning such as those summarized in the chart above.

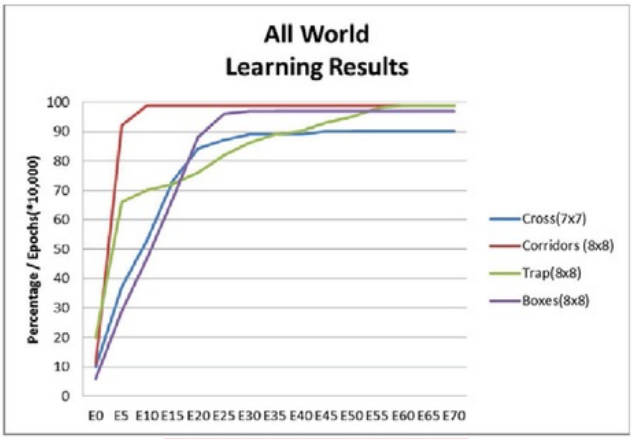

Fig. 2.16 All Worlds Learning Results

The graph above is an average yield of the whole arena used in the study of reinforcement learning using either agent or not. These results once told that there are no outstanding changes by using a combination of softmax policy action by learning method and greedy SARSA and Q-Learning.

The results showed that different types of arena shows a different performance from one arena to another arena. It is known that the difference is one arena to another arena is the number and location of obstructions (rocks) used by the agent to avoid the threat.

From information known about each type of area, obtained a preliminary estimate that the arena has a number of stone most agents have the opportunity to acquire for best performance. Conversely, arena rock that has a little amount is expected to get a less good performance.

From the results of this study also demonstrated that the location of obstructions (stones) in determining the performance can be obtained by the agent. For example agent Boxes (8x8) has a number of rock 20 (ranked the second highest compared to other arenas). But getting the worst performance on the agent's early learning. However, the placement location of the barrier on the arena Boxes (8x8) form a closed space in the center of the arena and just to give way in and out at the bottom of the course provides a great opportunity for the agent to reach the maximum value of 97% in the end.

Experiments have been done to give the facts about the best views of the type of agent used arena. With a length of time ignore the lessons provided by the agent, then obtained the following results:

Table 2.1Stone Comparison

Peringkat	*World*	Jumlah Batu
1	Corridors (8x8)	32
2	Trap(8x8)	16
3	Boxes(8x8)	20
4	Cross(7x7)	9

Results obtained from the following table is true that the arena Corridors (8x8) which has the most number of stones was ranked the best performance in terms of the maximum value that can be achieved. Instead, Cross (7x7) which has the least number of stones was ranked worst in terms of maximum performance. However, please note that the arena Boxes (8x8) has a number of stone more than the arena Trap (8x8). From the table we get, arena Trap (8x8) is superior Boxes (8x8) although the number of arena rock in the Trap (8x8) is less than the arena Boxes (8x8). From this it can be said that the large number of stones is not necessarily determine the maximum performance from agent performance.

Experiments have been carried out in four different types of arenas. Among them are: Cross (7x7), Corridors (8x8), Trap (8x8), and Boxes (8x8). From the experimental results obtained that the agent in Corridors (8x8) get the performance arena of learning faster than others. Agent in Corridors (8x8) has the fastest learning to achieve the maximum value of 99% in Epoch 100.000. This is achieved because this type of arena Corridors (8x8) has many stone barriers that can be used by the agent to avoid the threat.

Experiments conducted in Boxes (8x8) have a poor performance compared to other arenas in the beginning. But when learning reached 200.000 Epoch, agent on the performance of Cross (7x7) have declined in performance compared with the agent in Boxes (8x8). Achievement of the maximum that can be generated by the agent on the Cross (7x7) only reached a maximum percentage at 90% at 450.000, followed by the achievement Epoch maximum that can be generated by the agent in Boxes (8x8) which is 97% in Epoch 300.000.

Agent's performance on the arena Trap (8x8) categorized the second best after the agent Corridors (8x8) at first. But when it reaches the Epoch 150.000, agent Trap (8x8) have increased performance long enough to reach its maximum value compared with the agent Cross (7x7) and Boxes (8x8). Agent Trap (8x8) can reach a maximum point of 99%, equivalent to the best agent, Corridors (8x8), although it takes time to get the final value at the latest stable compared to other arenas agent, the Epoch 600.000.

5. Summary

The reason this research is to understand the methods of artificial intelligence in the game more in the reinforcement learning as one of the existing approaches in machine learning. To gain this understanding, it is necessary tool for the implementation of applications that support agent can be administered as a subject of learning reinforcement learning itself. In this study a methodology to take advantage of learning by reinforcement learning algorithms Q-Learning and Sarsa, complete with action selection policies for the implementation of measures to balance between exploitation and exploration of the agent. This methodology will then be tested through experiments on four arena (World) are different.

1. The first experiment is testing the ability of agents to act in the arena Cross (7x7). On this issue, without any learning agent is able to achieve a success rate of 49%. Meanwhile, after learning provided in accordance with the methodology used can achieve a success rate of 90%.
2. The second experiment is testing the ability of agents to act in the arena Corridors (8x8). On this arena, without any learning agent is able to achieve 61% success rate. Meanwhile, after a given study were Able to Achieve perfect success rate is 99%.
3. The third experiment carried out in the arena Trap (8x8). Agent managed to achieve a success rate of 48%. Performance of agents can achieve maximum success rate of 99% after the learning process.

To improve the speed performance of the agent in the learning environment is necessary improvements to existing algorithms techniques. The first problem is how to increase the success rate of agents in

performing its purpose in the arena / environment has to offer. On this issue, the proposed methodology successfully obtains good results. The second problem is the speed of the agent doing the learning, trial-and-error, exploitation and exploration. On this issue, the proposed methodology gave poor results. Here are some suggestions aimed at improving the results:

1. Carried out tests on each parameter value to determine how much influence each variable in the support agent to accomplish its purpose.
2. The use of engineering algorithms that can shorten the learning time for the agent in the learning environment. So the agent can obtain a level of success in a shorter acquisition time compared to methods used right now.

References

[1]Aha DW, Molineaux M and Ponsen M. *Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game*, In Proceedings of International Conference onCase-Based Reasoning, Chicago, USA , 23-26 August2005, pp. 5-20.

[2] Fang YP, Ting IH. *Applying Reinforcement Learning for the AI in a Tank-Battle Game*. JOURNAL OF SOFTWARE, VOL. 5, NO. 12, DECEMBER 2010.

[3] Sutton RS, Barto AG. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[4] Eden T, Knittel A, Uffelen RV. *Introduction to Reinforcement Learning*. http://www.cse.unsw.edu.au/~cs9417ml/RL1/index.html

[5] Kok E. *Adaptive reinforcement learning agents in RTS games*. Intelligent System group, University Utrecht, TheNetherlands; 2008.

[6]Fang, Y.-P., & Ting, I.-H. (2010). Applying Reinforcement Learning for the AI in a Tank-Battle Game. Journal of lSoftware, Vol. 5, No. 12, December 2010.

[7] Mićić, A., Arnarsson, D., & Jónsson, V. (2011). Developing Game AI For The Real-Time Strategy Game Starcraft. Research Report Spring 2011, Reykjavik University.

[8] Takadama, K. & Fujita, H. Lessons learned from comparison between Q-learning and Sarsa Agents in Bargaining Game. NAACSOS (North American Association for Computational Social and Organizational Science) Conference 2004

Check Game artificial intelligence based using reinforcement learning.pdf

ORIGINALITY REPORT

97%	40%	97%	28%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Agung, Albertus, and Ford Lumban Gaol. "Game Artificial Intelligence Based Using Reinforcement Learning", Procedia Engineering, 2012. Publication	96%
2	eprints.uthm.edu.my Internet Source	1%

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		